IRON: Inverse Rendering by Optimizing Neural SDFs and Materials from Photometric Images

Kai Zhang\textsuperscript{1} Fujun Luan\textsuperscript{2} Zhengqi Li\textsuperscript{3} Noah Snavely\textsuperscript{1}
\textsuperscript{1}Cornell University \textsuperscript{2}Adobe Research \textsuperscript{3}Google Research

Abstract

We propose a neural inverse rendering pipeline called IRON that operates on photometric images and outputs high-quality 3D content in the format of triangle meshes and material textures readily deployable in existing graphics pipelines. Our method adopts neural representations for geometry as signed distance fields (SDFs) and materials during optimization to enjoy their flexibility and compactness, and features a hybrid optimization scheme for neural SDFs: first, optimize using a volumetric radiance field approach to recover correct topology, then optimize further using edge-aware physics-based surface rendering for geometry refinement and disentanglement of materials and lighting. In the second stage, we also draw inspiration from mesh-based differentiable rendering, and design a novel edge sampling algorithm for neural SDFs to further improve performance. We show that our IRON achieves significantly better inverse rendering quality compared to prior works.

1. Introduction

Inverse rendering—the reconstruction of shape and appearance of real-world objects from a set of 2D input images—can enable accessible, high-quality digitization of our world. One way to formulate this problem is as the inversion of rendering algorithms used in computer graphics. Recent advances in graphics have led to fully differentiable Monte Carlo path tracing methods for jointly optimizing geometry and BRDFs represented as triangle meshes and material textures. However, meshes can be difficult to optimize, because it is non-trivial to modify their topology or maintain regularity during optimization. On the other hand, recently developed neural representations for shape \cite{mescheder2018occupancy, chang2019flame} and radiance fields \cite{park2019deepsdf, park2019deepsdf++} demonstrate impressive success in view synthesis \cite{park2019deepsdf} and shape reconstruction \cite{li2020occgan, li2020occupancy, lombardi2020neurokit, min2020neural, sun2020mstnet} tasks. But these representations entangle material and lighting, and cannot be directly used for applications like relighting or material editing. Recent methods for decoupling neural radiance fields are either limited to simple shapes \cite{chen2020dncml, zhao2020dncml}, or have neural components \cite{chen2021hacml, zhang2021compositing} that are incompatible with standard 3D rendering and editing tools.

We address the inverse rendering problem with the objective of embracing both the flexibility and compactness of neural representations, and the convenience of meshes with material textures in downstream applications. We present a pipeline we call IRON—Inverse Rendering by Optimizing Neural scene components, including neural signed distance fields (SDF) and neural materials. IRON performs an inverse rendering optimization starting from multi-view images captured by co-locating a flashlight with a moving...
camera [4, 5, 21, 24] (called photometric images in recent work), while allowing for the export of the optimized 3D content in mesh and material textures readily usable by traditional graphics renderers and AR/VR applications, as shown in Fig. 1. At the core of IRON are four compact neural networks representing the neural SDF and materials, which we optimize with a hybrid optimization scheme: first, a volumetric radiance field optimization to recover object topology, followed by physics-based surface rendering to refine geometric details and disentangle materials and lighting. We demonstrate superior inverse rendering quality over baseline methods targeting photometric images.

In addition, recent differentiable rendering methods in the graphics community emphasize the importance of carefully considering occlusion boundaries when performing inverse rendering with meshes, e.g., with special edge sampling algorithms that compute accurate gradients at such edges [2, 8, 17, 18, 28, 40]. In addition to our system as a whole, we propose an edge sampling algorithm defined for neural SDF representations, rather than meshes as in prior work. We show that our new edge sampling algorithm significantly improves reconstruction quality around edges.

Contributions. To summarize, our key contributions are:

- IRON, a neural inverse rendering pipeline for high-quality reconstruction of object shape and spatially varying materials, outperforming existing methods for photometric images.
- A hybrid optimization scheme for neural SDFs and materials that first optimizes a volumetric radiance field then performs edge-aware physics-based surface rendering for improved performance and compatibility with meshes and material textures.
- An edge-aware rendering optimization featuring a novel edge sampling algorithm that generates unbiased gradient estimates for better optimizing neural SDFs.

2. Related work

Neural reconstruction and view synthesis. Neural shape [22, 31] and appearance [29] representations based on Multi-layer Perceptrons (MLP) have been of recent interest, due to their compactness and representation power. Many works apply these representations to view synthesis and/or 3D reconstruction from multiple images, including NeRF [23], DVR [26], and IDR [39]. While NeRF yields view synthesis results of remarkable quality on complex scenes, its volume rendering nature leads to reduced quality of estimated surface geometry. In contrast, DVR and IDR, which utilize surface rendering, can reconstruct high-quality surface geometry, but are limited to relatively simple scenes compared to NeRF. Hence, UNISURF [30], VoSDF [38], and NeuS [34] sought to combine the benefits of both volume- and surface-based rendering by considering surfaces as defining volumes near the surface. Our work also utilizes neural shape and appearance, but rather than encoding appearance as a surface light field [36] that entangles lighting and materials, we adopt a physics-based surface shading model within an inverse rendering framework. Our pipeline can output meshes and materials readily importable by existing graphics pipelines, e.g., Blender [9], for fast raytracing, object insertion, relighting, and material editing.

Mesh-based differentiable rendering. Mesheres are widely used in graphics pipelines and game engines. There has been a surge of recent interest in fully-differentiable forward rendering methods [2, 8, 17, 18, 20, 27, 28, 40], which enable joint optimization of shape, material, and camera parameters from images [21]. One challenge is proper derivative computations at depth discontinuities (called edge derivatives) that arise when rendering a mesh to an image. In addition, meshes are not friendly for shape optimization, due to difficulties posed in changing their topology and avoiding self-intersections [23]. In contrast, signed distance field (SDF) representations commonly used in neural geometry methods [31] parametrize the surface as the zero level set of a continuous SDF (instantiated in neural methods by an MLP), alleviating these challenges. However, we observe that, like meshes, optimizing a neural SDF via differentiable surface rendering also requires computing both interior derivatives and edge derivatives with respect to neural network weights. IDR [39] only uses interior derivatives, leading to poor performance around edges. We introduce a method to estimate edge derivatives for better optimization of neural SDFs.

Inverse rendering from multiple images. Given multiple images of a scene, inverse rendering methods seek to recover the shape, material and lighting that best explain the observed images. Prior work considers several capture scenarios, including: 1) static scene, static environmental lighting, moving camera [6, 41, 43], 2) rotating scene, static environmental lighting, fixed camera [11, 37], and 3) static scene, flashlight illumination co-located with moving camera [4, 5, 21, 24] (in some works called photometric images). Our work addresses the third case of flashlight photography because it simplifies the physics-based rendering module, while enabling high-quality inverse rendering results. Using neural SDFs and materials, we show improved results compared to baselines, while also creating easily deployed mesh and texture outputs.

3. Method

Assumptions. We focus on opaque objects, and consider transparent and translucent objects outside the scope of our work. We also assume that the input photometric images are captured using collocated flashlight illumination without ambient light. Finally, for efficiency, we ignore shadows (which are minimal in practice due to the collocated flashlight) and global illumination effects like interreflections.
Under these assumptions, and provided with an input set of photometric images, our IRON system optimizes for neural shape and material representations consisting of four compact MLPs with positional encoding [23, 32]:

**Neural SDF** $S_{\Theta_{s}} : x \rightarrow (S, f)$ maps a 3D location $x$ to an SDF value $S$ and a 256D local geometric feature descriptor $f$, as in recent works [30, 34, 39]; this feature descriptor can then be fed into our neural material networks.

**Neural diffuse albedo** $\beta_{\Theta_{\beta}} : (x, n, f) \rightarrow \beta$ outputs the diffuse albedo $\beta$ at location $x$ given surface normal $n$ and feature descriptor $f$. This function plays a dual role in our optimization. In our first optimization phase, we treat $\beta$ as a neural radiance field, and include view direction as an additional parameter (in place of the second $n$). In the second phase, we constrain $\beta$ to represent diffuse albedo in our edge-aware physics-based surface rendering. Our neural diffuse albedo MLP utilizes the surface normal and local geometric feature descriptors as in prior work [30, 34, 39].

**Neural specular albedo** $\kappa_{\Theta_{\kappa}} : (x, n, f) \rightarrow \kappa$ encodes spatially-varying specular albedo $\kappa$.

**Neural roughness** $\alpha_{\Theta_{\alpha}} : (x, n, f) \rightarrow \alpha$ encodes the spatially-varying specular roughness. Small values indicate shiny surfaces, and large values less shiny.

We optimize the MLP weights $\Theta_{s}, \Theta_{\beta}, \Theta_{\kappa},$ and $\Theta_{\alpha}$ and a scalar light intensity $L$ in a two-stage optimization scheme. In the first stage, we optimize neural SDF $S_{\Theta_{s}}$ and diffuse albedo $\beta_{\Theta_{\beta}}$ by treating $\beta_{\Theta_{\beta}}$ as a volumetric radiance field. This phase is designed to recover correct object topology and serves as an initialization for the second phase. In the second phase, we refine geometric details and factorize materials from lighting by jointly optimizing neural SDF $S_{\Theta_{s}}$, neural materials $\beta_{\Theta_{\beta}}, \kappa_{\Theta_{\kappa}},$ and $\alpha_{\Theta_{\alpha}},$ and light intensity $L$ via an edge-aware physics-based surface rendering method.

### 3.1. Volumetric radiance field rendering

In the first stage, we optimize $\beta_{\Theta_{\beta}}(x, n, d, f)$ as a view-dependent neural radiance field (by substituting view direction $-d$ for the second $n$). Hence, we perform volumetric radiance field rendering of the neural SDF $S_{\Theta_{s}}$, and colors $\beta_{\Theta_{\beta}}(x, n, d, f)$ as in [34] in order to harness the power of volume rendering in recovering correct object topology [23, 30, 34, 38], i.e., the correct number and location of holes in the geometry. If we do not perform this initial optimization stage and instead optimize for surface rendering from scratch, we found that the optimization frequently diverges unless object segmentation masks are provided, as observed by [30], and often gets stuck in local minima with incorrect topology. At the same time, the volumetric nature of this stage is inconsistent with our goal of producing shape and materials that are compatible with the mesh-based rendering paradigm used in existing graphics pipelines. This motivates the second optimization stage where we perform edge-aware physics-based surface rendering.

### 3.2. Edge-aware physics-based surface rendering

After solving for a volumetric radiance field, we perform a second, full inverse rendering stage that jointly optimizes neural SDF $S_{\Theta_{s}}$, and neural materials $\beta_{\Theta_{\beta}}, \kappa_{\Theta_{\kappa}},$ and $\alpha_{\Theta_{\alpha}},$ from photometric images. This stage has two key components: differentiable physics-based shading and edge-aware surface rendering.

**Physics-based shading.** As our photometric image inputs have co-located flashlight and camera, the light direction is aligned with the view direction across pixel locations. Hence, we can simplify the rendering equation [16] as:

$$L_{o}(\omega_{o}, x) = \int_{\Omega} L_{i}(\omega_{i}, x)f_{r}(\omega_{o}, \omega_{i}, x)(\omega_{i} \cdot n)d\omega_{i}$$

$$\approx L_{i}(\omega_{o}, x)f_{r}(\omega_{o}, \omega_{i}, x)(\omega_{o} \cdot n),$$

where $L_{o}, x, n, \omega_{o}, \omega_{i}, L_{i}, f_{r}$ are observed light, surface location, surface normal, light direction, view direction, incident light and BRDF, respectively. We model the white flashlight as a point light source as in [21, 24]:

$$L_{i}(\omega_{i}; x) = \frac{L}{\|x - o\|^2},$$

where $L$ is a scalar light intensity and $o$ is the light location (same as the camera location in our co-located capture setup). The denominator models the inverse-square fall-off in intensity. For BRDF $f_{r}$, we use the GGX model [33], whose parameters at location $x$, diffuse albedo $\beta$, specular albedo $\kappa$ and roughness $\alpha$, are encoded in our neural materials $\beta_{\Theta_{\beta}}, \kappa_{\Theta_{\kappa}},$ and $\alpha_{\Theta_{\alpha}},$ respectively.

From Eqns. 2 and 3, we observe that gradients from the rendered image $L_{o}(\omega_{o}; x)$ must back-propagate through surface location $x$ and surface normal $n$ to the shape parameters, and through the BRDF $f_{r}$ to the material parameters.

**Edge-aware surface rendering.** Inspired by recent advances in mesh-based differentiable rendering [2, 18, 21, 28,
Assuming known color, prior neural surface rendering methods [26, 39] fail to deform a small sphere to a target large sphere due to lack of edge handling. The shading at an edge pixel involves the combination of the shading at disconnected surface pieces identified using subpixel localization of edges. We locate edge points in 3D by walking on the zero level set of the neural SDF, then project 3D edge points into 2D for subpixel edge localization. Figure 2. Illustration of edge-aware surface rendering for neural SDFs. (a) Existing neural surface rendering methods ignore geometric discontinuities, making it difficult to deform neural SDFs to match silhouettes even for simple objects. (b) Geometric discontinuities are introduced by edge pixels where multiple depth values are present in a single pixel, motivating our proposed edge sampling algorithm. (c) Our method can localize subpixel-accurate edges for neural SDFs enabling correct shading calculations at edge pixels.

Figure 3. Demonstration of edge-aware surface rendering for neural SDF. To render the image (a), we first run sphere tracing to get the depth (b) and surface normal (c) corresponding to the pixel centers. We then localize subpixel-accurate edges by walking on the surface; the projected walk direction (d) is color-coded as shown in the inset color wheel. Detected edge pixels are shown in red in (e). We shade non-edge pixels as in existing neural surface rendering methods [26,39], but shade edge pixels by accounting for the geometric discontinuity.

We address this issue via a novel edge sampling algorithm tailored for neural SDFs. Our algorithm has three steps: 1) localize subpixel-accurate 2D edges by detecting 3D edge points that are then projected to the 2D image, 2) reparametrize edge points such that they can back-propagate gradients to the neural SDF in an auto-differentiation framework, and 3) compute the shading color for edge pixels.

In step 1, we start from ray-surface intersections found by sphere tracing the center rays at each pixel location [13] (Fig. 3(b,c)), and walk on the surface along the direction defined in line 6 of Alg. 1 and illustrated in Fig. 2(c) until reaching a 3D edge point or a max number of steps [3, 7]. We visualize projected walk directions in Fig. 3(d). For the sake of efficiency, we only do the surface walk process for the ray-surface intersections at depth discontinuity pixels in order to reduce the number of evaluations of the neural SDF; we identify such depth discontinuity pixels as ones with depth Sobel gradient magnitude above a certain threshold $\tau$. We then project detected 3D edge points to image space, producing both subpixel-accurate edge locations and an edge mask marking pixels containing edges (Fig. 3(e)). We also obtain 2D edge normal directions by projecting the 3D surface normals at these edge points to 2D.

In step 2, we reparameterize the edge points’ locations $x$ to make them differentiable with respect to network weights of the neural SDF. We observe that the differentiable ray-surface intersection equation (Eq. 4) for interior points in [26, 39] only captures how perturbations to neural SDF weights move the ray-surface intersection along the camera ray, while for edge points, we care about their movement along the surface normal. Hence, to reparameterize edge...
points correctly, we propose to replace the viewing direction $o - x$ ($o$ is the camera origin, $x$ is a surface point) in Eq. 4 with the surface normal $n$, and arrive at Eq. 5.

$$x_{\Theta_s} = x - \frac{o - x}{n^T(o - x)}S_{\Theta_s}(x),$$

$$x_{\Theta_s} = x - \frac{n}{n^Tn}S_{\Theta_s}(x) = x - nS_{\Theta_s}(x).$$

(4)

(5)

We show the correctness of our edge point reparametrization in the supplemental material.

In step 3, we compute the shading at each edge pixel. Consider the edge pixel in Fig. 2(b), and let the green projected edge point have subpixel coordinates $[u, v]$, and the green projected surface normal be $[du, dv]$. Then the black center of this edge pixel has coordinates:

$$[u_c, v_c] = \left[\text{floor}(u), \text{floor}(v)\right] + 0.5.$$

(6)

We approximate each square pixel footprint using a circle of radius $\frac{\sqrt{2}}{2}$ pixels centered at $[u_c, v_c]$. We then pick 2D locations, labeled A and B, on the circle on either side of the edge. We raytrace and shade the two selected 2D locations. Let us denote the shaded colors as $C_A$ and $C_B$, respectively. We linearly combine $C_A$ and $C_B$ with weights proportional to the two segment areas separated by the edge. Suppose the fraction of segment area on the same side as A is $w_A \in [0, 1]$:

$$\alpha = 2 \cdot \arccos(\sqrt{2} \cdot [du, dv][u - u_c, v - v_c]^T),$$

$$w_A = 1 - \frac{1}{2\pi} \cdot (\alpha - \sin\alpha),$$

(7)

(8)

then our predicted edge pixel color is:

$$C = w_A C_A + (1 - w_A) C_B.$$

(9)

In Eq. 9, gradients from $C$ can back-propagate through $w_A, C_A$, and $C_B$ to the parameters of our neural SDF and materials, and through the variable $w_A$, we correctly model how tiny perturbations of the 3D edge point’s location along the surface normal direction affect the edge pixel color.

### 3.3. Training and testing

Given multi-view photometric images, we optimize our neural SDF and materials using the following loss:

$$L = L_2(\text{pyramid}(\hat{I}), \text{pyramid}(I))$$

$$+ 1 - \text{SSIM}(\hat{I}, I)$$

$$+ \lambda_1 \cdot ||\nabla x S - 1||_2^2$$

$$+ \lambda_2 \cdot \text{max}(\text{roughness}(x) - 0.5, 0),$$

(10)

(11)

(12)

(13)

where Eq. 10 is the $L_2$ loss on Gaussian pyramids of the predicted image $\hat{I}$ and groundtruth image $I$, Eq. 11 is the SSIM loss [35], Eq. 12 is the eikonal loss [10, 12] enforcing the validity of the SDF, and Eq. 13 is the roughness range loss encouraging the estimated roughness to stay below 0.5. $\lambda_1, \lambda_2$ are loss weights.

Once training is complete, we convert the neural SDF and materials to a triangle mesh and texture map for deployment in the standard graphics pipeline. We first extract a mesh from the optimized neural SDF using the marching cube algorithm [19]. We then use the Blender Smart UV Project tool [9] to compute a reasonable per-vertex uv mapping. Finally, to fill the material texture images, we densely sample points on our triangle meshes with trilinearly interpolated uv coordinates, then query the material networks at the sampled surface points, and finally splat the per-point material parameters to the texture images using their interpolated uv coordinates.

### 4. Evaluation

We perform extensive experiments to validate IRON.
Figure 6. Comparison of recovered geometry on synthetic (left) and real data (right) with baselines DRV [4] and PSDR [21]. On synthetic data, we achieve the highest geometric reconstruction accuracy among the inverse rendering methods designed for flashlight photography. For real data, we show a captured photo for reference due to lack of ground truth geometry.

Figure 7. Qualitative comparison of generalization to novel co-located flashlight relighting using both synthetic (top two rows) and real (bottom two rows) data. Our IRON method produces visually more faithful matches in terms of geometric and texture details compared to DRV [4] and PSDR [21]. Results of the recent neural surface reconstruction work NeuS [34] are also included for reference.

4.1. Optimizing neural SDFs to fit single image

We design a controlled experiment to illustrate the key effect of our edge sampling algorithm: allowing the image loss to lead to in-plane neural SDF deformations. In this experiment, we assume that a single image of a 3D object with known constant color is given as input to optimize a neural SDF through an image loss. To solve this problem, the neural SDF must be deformed such that its silhouette matches the ground truth when viewed from the input viewpoint. But as shown in Fig. 4, the prior neural surface rendering
method IDR [39] fails to accomplish this task due to a lack of edge pixel handling. Our edge-aware surface rendering method addresses this problem, converging to a valid solution. Although mesh-based differentiable rendering methods can also handle silhouettes [40], we observe degraded mesh quality over the course of optimization without intermediate remeshing, due to the otherwise fixed mesh topology. In contrast, neural SDFs do not suffer from such problems.

4.2. Inverse rendering from photometric images

We now evaluate methods on the task of inverse rendering from multi-view photometric images.

Datasets. We create a synthetic dataset consisting of 9 objects: dragon, buddha, camera, monk, kettle, duck, pig, sneaker, and bagel. Each object is rendered from 200 randomly sampled viewpoints using the Mitsuba path-tracing renderer [14] to form the training data. We co-locate a point light source with the camera to actively illuminate the object without other light sources. We render test data consisting of 100 images under novel co-located flashlight illumination, and another 100 images under novel natural environmental illumination. For real-world data, we use 5 object captures from DRV [5]: dragon, pony, girl, tree, and triton. The data is acquired using co-located flashlight setup in a dark environment. We randomly choose 70% of the images for training and use the remaining 30% as test images for evaluating generalization to novel view with co-located lighting.

Baselines. We compare with two inverse rendering methods that use densely captured photometric images: the volume-based DRV [5] and the mesh-based PSDR [21] methods. We did not compare with the mesh-based method of Nam et al. [24] due to the lack of open-source code. In addition, PSDR has shown superior reconstruction quality compared to this method.

Discussion. As shown in Tab. 1 and Figs. 6 and 7, we outperform state-of-the-art inverse rendering baselines by a large margin in terms of geometric accuracy and generalization to novel co-located lighting on synthetic data, and produce fewer artifacts like blurry textures on real data. The
Figure 9. Ablations on our edge sampling algorithm using the synthetic dragon and real pony data. Without edge sampling, neural surface rendering performs poorly around edge regions, as shown by the error maps and surface normals.

Ablations on edge sampling. We test removing the edge sampling method from IRON. As shown in Fig. 9, the resulting reconstructions have much poorer quality around edge regions, e.g., the dragon horn and pony nose. This poor quality is due to the missing edge derivatives in existing neural surface rendering methods [26, 39].

Ablations on loss functions. We find that using the SSIM loss significantly improves the sharpness of reconstructions.

Figure 10. Ablations on our loss functions using the real dragon data. Without patch-based SSIM loss, our synthesized image is quite blurry, while VGG loss results in spurious texture details. The roughness range loss fixes issue of over-estimating roughness.

On the real dragon data, we first try removing SSIM loss from our pipeline. Then we also try replacing SSIM loss with VGG loss [15], which is widely used in view synthesis tasks. As shown in Fig. 10, our proposed loss functions lead to the sharpest and most plausible material reconstructions.

5. Conclusion

In this work, we have presented our new IRON pipeline specialized in high-quality inverse rendering from photometric images. In contrast to mesh-based differentiable rendering, IRON adopts neural SDFs and materials as representations for ease of optimization in the form of both volumetric radiance rendering and edge-aware physics-based surface rendering, and preserves the convertibility to meshes and material textures for the sake of downstream applications.

Limitations and future work. There are limitations to be resolved in future work. First, the use of photometric images leads to a more involved data capture process, although such images simplify inverse methods because of the known single point light and minimal shadows. Future work can explore extensions of our work to the combination of flashlight and ambient illumination. Second, we do not model multiple bounces of light. This can lead to material estimation errors in concave regions that feature significant interreflection. Future directions include devising computationally-efficient global illumination rendering algorithms for neural SDF representations. Third, our current BRDF model assumes opaque surfaces, and thus we do not expect our method to work well on transparent and translucent objects that feature significant refraction and subsurface scattering.

Acknowledgements. This work was supported in part by the National Science Foundation (IIS-2008313), and by funding from Intel and Amazon Web Services. We also thank Sai Bi for providing their code and data.
References


[18] Tzu-Mao Li, Miika Aittala, Frédéric Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. SIGGRAPH Asia, 37(6), 2018. 2, 3


